



基于信息增益的 Hadoop 瓶颈检测算法

谭造乐¹, 郝志峰¹, 蔡瑞初¹, 肖晓军², 卢宇²

(1. 广东工业大学计算机学院, 广东 广州 510006;

2. 广州优亿信息科技有限公司, 广东 广州 510630)

摘要: 当今, Hadoop 已经成为了大数据存储和大数据挖掘的主要平台。虽然 Hadoop 平台通过分布式的机器集群来实现高性能的并行计算, 但由于其由廉价主机组成, 故当集群负载增大时, 便不可避免地在某机器上出现瓶颈。针对此问题, 提出一种基于信息增益的瓶颈检测算法, 该算法通过计算各个资源的信息增益来检测集群的瓶颈资源。实验证明了该瓶颈检测算法具有可行性。

关键词: 大数据; Hadoop; 信息增益; 瓶颈检测

中图分类号: TP391

文献标识码: A

doi: 10.11959/j.issn.1000-0801.2016203

Hadoop bottleneck detection algorithm based on information gain

TAN Zaole¹, HAO Zhifeng¹, CAI Ruichu¹, XIAO Xiaojun², LU Yu²

1. School of Computers, Guangdong University of Technology, Guangzhou 510006, China

2. Guangzhou Usease Information Technology Co., Ltd., Guangzhou 510630, China

Abstract: Hadoop has become a major platform for big data storage and large data mining nowadays. Although Hadoop platform achieves high performance parallel computing through a distributed cluster of machines, the bottlenecks will inevitably appear on a machine when cluster load increases, because the cluster is composed of inexpensive host. Aiming at this problem, a bottleneck detection algorithms based on information gain was proposed. The algorithm detected cluster's bottlenecks resource by computing the information gain of each resource. The experiments show that the bottleneck detection algorithm is feasible.

Key words: big data, Hadoop, information gain, bottleneck detection

1 引言

而今互联网迅猛发展, 无论是网民规模还是网页规模均呈现出爆炸式增长。截至 2015 年 12 月, 我国网民规模达 6.88 亿人, 而网页数量更是高达 2 123 亿个^[1]。随之而来的便是网络数据的指数增长。故传统的数据处理框架已经难以应对当今海量数据的处理。而 Hadoop 的出现可以

很好地解决海量数据分析处理的难题。其通过分散数据集, 使计算能在分布式集群里高度并行地执行。Hadoop 使得在廉价机器集群上处理大数据集得以实现。

虽然 Hadoop 很好地解决了大数据的存储和处理问题, 但随着数据的多样化以及企业对数据处理的需求更严格, Hadoop 在性能方面也呈现出了缺陷。国内有不少学者针对 Hadoop 优化方面进行了研究。如参考文献[2]通



过 mapFile、HBase 等 Hadoop 生态系统的相关组件,对小文件处理进行优化;参考文献[3]在底层的调度器方面对 Hadoop 进行优化;而参考文献[4-6]则通过 MapReduce 参数调整优化了提高 Hadoop 性能。但这些都只局限于优化,而忽视了具体瓶颈检测方面的研究。针对系统瓶颈检测,参考文献[7]用机器学习的方法来计算 SLO 满意度并对系统瓶颈进行检测;参考文献[8]提出一种基于决策树的瓶颈检测方法,但其局限于针对运行 Web 服务的平台进行检测。基于此,本文提出一种基于资源信息增益的瓶颈检测算法,通过对集群性能进行监控,检测瓶颈的出现。然后通过各个资源使用率和评价指标响应满意度(response satisfaction, RS)的样本计算各种资源的信息增益,选取信息最大的资源作为集群的瓶颈资源。在确定瓶颈资源后便可以针对该瓶颈资源进行相应调节措施,从而对集群进行优化。实验通过 Wordcount 和文本聚类这两类测试用例来验证瓶颈检测算法的可行性。通过增加 Wordcount 输入文件大小和文本聚类数据量来不断增加负载,然后使用本文的检测算法来确定瓶颈资源。实验证明,应用本文的检测算法后的集群性能比优化前有很好的提升。可见本文提出的瓶颈检测算法能有效地检测出具体的瓶颈资源,为集群调整优化提供了坚实的基础。

2 Hadoop 平台架构

Hadoop 是开源社区 Apache 根据谷歌公布的分布式文件系统(GFS)技术^[9]和 MapReduce^[10]编程模型而开发出的分布式处理框架。其不仅使用一种廉价的方式对海量数据进行存储,还提供高效可靠且可扩展的数据处理机制。其中,分布式文件系统 HDFS (Hadoop distributed file system)和 MapReduce 是 Hadoop 的核心部分。HDFS 为海量的数据提供了存储,MapReduce 则为海量的数据提供了计算。

HDFS 的流程架构如图 1 所示。其中, nameNode(主节点)扮演着管理者的角色,其保存了整个集群文件系统的所有元数据。这些元数据包含文件名、访问权限和各个块的位置,一般保存在内存中,从而保证快速的随机访问。而 dataNode(数据节点)是存储的基本单元。HDFS 是基于块结构的文件系统。其将单个文件拆分成固定大小的块,然后将这些块保存在数据节点中。同时数据节点会周期性地向主节点发送心跳信息和本地的块报告。当客户端需要读数据时,首先从主节点获取数据所在的位置信息,即哪些数据节点保存有要读数据的块。然后客户端与最近的数据节点进行通信,读取所需数据。

而 MapReduce 编程模型采用“分而治之”的思想。其运行机制示意如图 2 所示。mapper 任务调用用户自定义 map 函数后对中间结果进行局部排序,然后运行 combiner 对数据进行合并。当所有 mapper 任务都完成后, reducer 任务会接收到 mapper 的通知,然后通过 RPC 远程调用将 mapper 任务产生的 M 份属于自己的数据文件远程拉(pull)取到本地。当所有中间结果数据拉取成功后,采用归并排序来对中间结果 key 进行排序,然后调用用户自定义的 reduce 函数进行业务逻辑处理并输出结果。虽然 Hadoop 集群很好地解决了海量数据的分布式存储和并行计算问题,但当集群负载增大时,还是不可避免地会出现瓶颈,从而导致集群性能下降。

3 瓶颈检测算法

当集群运行出现瓶颈而没有及时检测出瓶颈资源时,不能采取相应优化措施,会严重影响整个集群的正常运行,这将大大降低集群性能,甚至使集群运行停滞。针对此,提出一种基于信息增益的瓶颈检测算法,通过将该算法模型部署到集群中,实时检测集群的性能。当在某节点

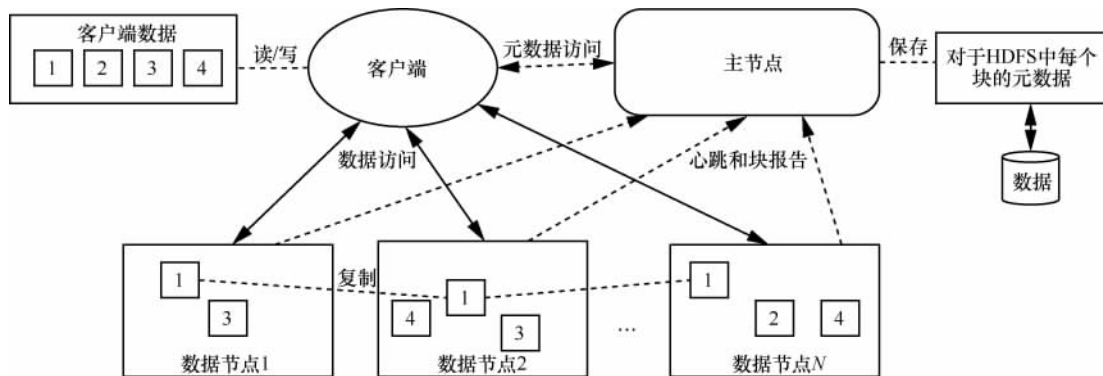


图 1 HDFS 的流程架构

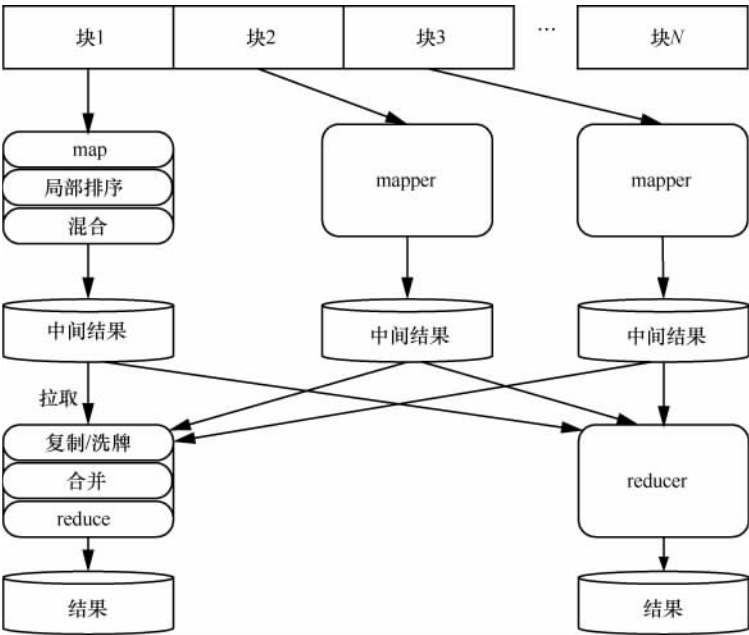


图 2 MapReduce 运行机制示意

中检测到瓶颈出现时，通过计算各种资源的信息增益,从而确定最可能成为瓶颈的资源。

3.1 性能评价指标

衡量一个系统性能最常见的指标有响应时间、吞吐量、资源使用率等,其中最直观的可能是资源使用率,因为一般瓶颈资源都是使用率比较高的资源。但资源的高使用率并不一定意味着该资源就是瓶颈资源。高使用率只是瓶颈资源的必要条件,而非充要条件。例如在 Linux 系统中,其总是试图让 CPU 尽量繁忙,从而最大化作业任务的吞吐量。而在 Hadoop 性能指标方面,Apache Eagle^[11],一个开源的分布式实时监控及预警框架主要使用下列指标来评价 Hadoop 作业性能是否异常。

- 作业进度缓慢。当作业在相当一段时间(15 min)内没有对 HDFS 或文件进行读写时,则认为作业运行出现异常。
- 作业用时过长。通过作业运行的耗时来评价 Hadoop 性能。当作业持续耗时超过 12 h,则认定作业性能差。
- reduce 数量过大。若单个作业 reduce 数量超过了 1500,则性能低下。

- 作业与历史记录比较,用时太久。通过采集 100 个历史样本,除掉 10%最长用时,计算中位数 P 和离差倍增数的 95 分位数 Dm 。当作业运行时间大于 $\max(P,Dm)$ 时,则认为性能差。
- CPU 平均每秒消耗字节数 (bytes consumed per CPU second)太低。CPU 平均每秒消耗字节数不低于 200 KB。
- 洗牌大小(shuffle size)。设定单个 reduce 对应的洗牌大小不可以超过一定阈值(10 GB)。

而根据 eBay 2015 年 5 月的数据,各指标比例见表 1。可见,大部分可以从作业运行进度和时间指标去评价 Hadoop 性能。基于此,本文使用响应满意度(RS)来评价集群性能。

$$RS=\begin{cases} 1, & T_m>T_s \\ \frac{T_m}{T_s}\times100\%, & T_m\leq T_s \end{cases} \quad (1)$$

其中, T_m 为期待最小响应时间, T_s 为系统响应时间。

3.2 资源信息增益

由于影响集群系统性能的资源有很多,且检测初期并不知道可能成为瓶颈的资源有哪些,故将可能导致系统瓶

表 1 各类指标比例

指标	作业进度	reduce 数量	CPU 每秒消耗字节数	作业用时与历史对比	作业用时	洗牌大小
比例	32%	24%	23%	15%	3%	2%



颈的资源都考虑进去。如 CPU、内存、硬盘 I/O 和网络带宽等。在数据预处理中,要对数据集进行离散化,即对资源使用率和响应满意度进行等级分类。其中,资源使用率分 5 类:verylow(非常低)(0~20%)、low(低)(20%~40%)、middle(中等)(40%~60%)、high(高)(60%~80%)、veryhigh(非常高)(80%~100%),而响应满意度分 3 类:low(低)(0~60%)、middle(中)(60%~80%)和 high(高)(80%~100%)。例如有样本:

$s = \{\text{CPU}:45\%, \text{mem}:25\%, \text{I/O}:13\%, \text{network}:66\%, \text{RS}:85\%\}$

则其经过离散化后,样本变为:

$s' = \{\text{CPU}:\text{middle}, \text{mem}:\text{low}, \text{I/O}:\text{verylow}, \text{network}:\text{high}, \text{RS}:\text{high}\}$

在数据挖掘中,使用 ID3 算法构建决策树时,通常会使用信息熵和信息增益来确定对目标分类影响最大的主要属性^[12]。本文也类似地通过资源的信息增益来确定与 RS 关系最密切的资源。首先需要计算数据集的信息熵。这里的信息熵可以理解为用于衡量系统复杂性的信息量。由上已经知道目标属性 RS 有 3 种可能等级,这里记为 $RS_i (i=1,2,3)$ 。假设在整个数据集 S 中, RS_i 出现的概率为 P_i ,则数据集 S 所含的信息熵为:

$$\text{entropy}(S) = - \sum_{i=1}^3 P_i \lg P_i \quad (2)$$

然后计算根据资源划分后样本子集的信息熵,即条件熵。假定资源 A 在数据集中存在 k 个不同的值,那么可以根据资源 A 将总样本集划分为 k 个样本子集 $\{s_1, s_2, \dots, s_k\}$,则按资源 A 划分后的样本子集信息熵为:

$$\text{entropy}_A(S) = \sum_{i=1}^k \frac{|S_i|}{|S|} \text{entropy}(S_i) \quad (3)$$

其中, $|S_i| (i=1,2,\dots,k)$ 为子集 S_i 的样本数, $|S|$ 为总样本数。而信息增益则是样本划分前后信息熵的差值,其代表着按资源划分样本后而导致期望熵的减少。按资源 A 划分总数据集 S 的信息增益为:

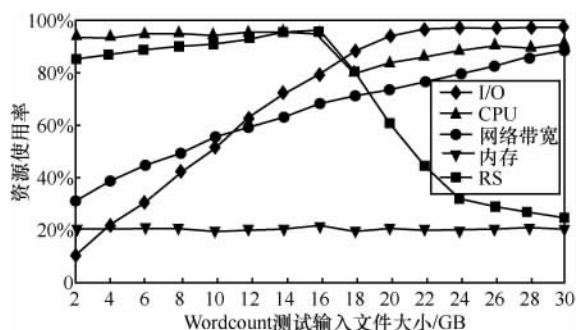
$$\text{gain}(S,A) = \text{entropy}(S) - \text{entropy}_A(S) \quad (4)$$

信息增益越大,则说明该资源对作业 RS 的影响就越大。故当集群系统的 RS 大幅度下降且降到一定阈值(如 60%)时,通过采集各个资源使用率与响应满意度样本集并离散化后,计算各个资源的信息增益。最后将具有最高信息增益的资源作为系统的瓶颈资源。然后针对瓶颈资源对集群参数进行调整优化,以缓解瓶颈问题。例如若检测到磁盘 I/O 是最可能造成集群出现瓶颈的资源,则可以通过增大 io.sort.mb 、 dfs.block.size 等参数,减少作业运行时的

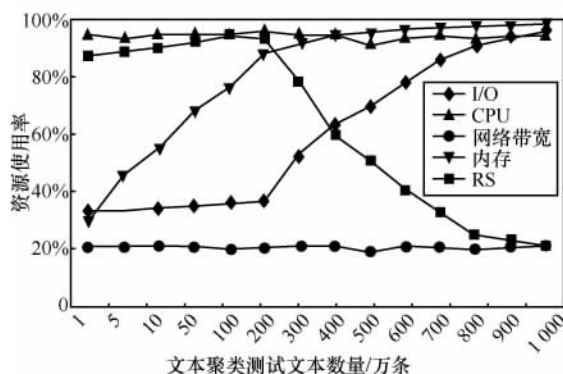
磁盘 I/O 操作。若网络带宽成为瓶颈资源,则可设置 $\text{map.output.compress}$ 参数对作业运行时中间结果数据进行压缩,以减少传输的带宽。

4 实验设计与分析

实验通过搭建 3 台机器的 Hadoop 集群来进行实验,其中,实验机器 CPU 为 Intel XEON 2.0,内存为 2 GB,硬盘 100 GB,操作系统为 Ubuntu 14.04。实验测试用例选用 Wordcount 和文本聚类,并通过不断增加输入文件大小和文本数量来测试集群的负载情况,然后记录不同负载下作业的 RS 及集群中各种资源利用率情况,再通过瓶颈检测算法计算具体的瓶颈资源。具体实验结果如图 3 所示。



(a) 资源使用率与 Wordcount 测试输入文件大小



(b) 资源使用率与文本聚类测试文本数量

图 3 资源使用率与响应满意度

可见,使用 Wordcount 测试用例时,当输入文件增加到 18 GB 以上时,作业的响应满意度开始大幅度下降。当集群负载持续增大时,其响应满意度已经开始逐渐趋于零。可见随着负载加大到一定程度,集群开始出现瓶颈。瓶颈的出现会导致集群性能大幅度下降且最后基本处于停滞状态。故针对图 3 中 Wordcount 测试用例的资源使用率 and 对应满意度数据,根据本文的瓶颈检测算法来检测瓶颈资源。由于内存(mem)资源一直处于低使用率状态,

故其不可能成为瓶颈资源。所以只考虑 CPU、网络带宽 (network) 和 I/O 这 3 种资源。首先对数据进行离散化, 离散化后数据见表 2。

表 2 样本离散化后数据

磁盘 I/O	CPU	网络带宽	响应满意度
非常低	非常高	低	高
低	非常高	低	高
低	非常高	中等	高
中等	非常高	中等	高
中等	非常高	中等	高
高	非常高	中等	高
高	非常高	高	高
高	非常高	高	高
非常高	高	高	中
非常高	非常高	高	中
非常高	非常高	高	低
非常高	非常高	高	低
非常高	非常高	非常高	低
非常高	非常高	非常高	低
非常高	非常高	非常高	低

然后根据样本计算各个资源的信息增益。首先计算整个数据集信息熵 $\text{entropy}(S)=1.399\ 6$, 然后计算基于网络带宽划分的子样本的信息熵 $\text{entropy}_{\text{net}}(S)=0.633\ 9$, 故可以得到 $\text{gain}(\text{network})=0.765\ 7$ 。同理可以计算出 $\text{gain}(\text{CPU})=0.126\ 8$, $\text{gain}(\text{I/O})=0.996\ 9$ 。从而可以判断集群瓶颈资源为 I/O。同理, 根据图 3 中文本聚类测试用例数据可以算出瓶颈资源为内存(信息增益为 0.409 9)。可见虽然在两个测试用例中, CPU 一直处于高使用率, 但其并非是造成瓶颈的主要资源。反观在 Wordcount 中, I/O 资源使用率开始时随着输入文件大小的增加, 其利用率保持一定的增长速率。但当输入文件大小达到一定数量时, 其使用率增长率开始趋于零, 与此同时, 作业的 RS 开始大幅度下降。可知 I/O 资源是导致 RS 下降的最主要原因, 故网络带宽是造成采集模块瓶颈的资源。而在文本聚类测试用例时, I/O 使用率在样本数量达到 300 万时突然由稳定开始上升, 这主要是因为当样本集过大时, 内存无法缓存所有数据, 需要将数据分批次缓存到内存中, 增加了磁盘 I/O 操作, 从而提高了 I/O 使用率。在 Wordcount 测试用例中, 将 `io.sort.mb` 参数从默认的 100 MB 提升到 200 MB, 增加排序的内存。同时增加 `dfs.blocksize` 参数, 将文件块的大小由 64 MB 增加到 128 MB, 从而减少磁盘 I/O 的操作。而文本聚类则将机器

内存由 2 GB 增大到 4 GB。调整优化后再进行上述测试用例实验, 得到 RS 对比如图 4 所示。可见经过瓶颈检测调优后的集群能有效地避免瓶颈出现。

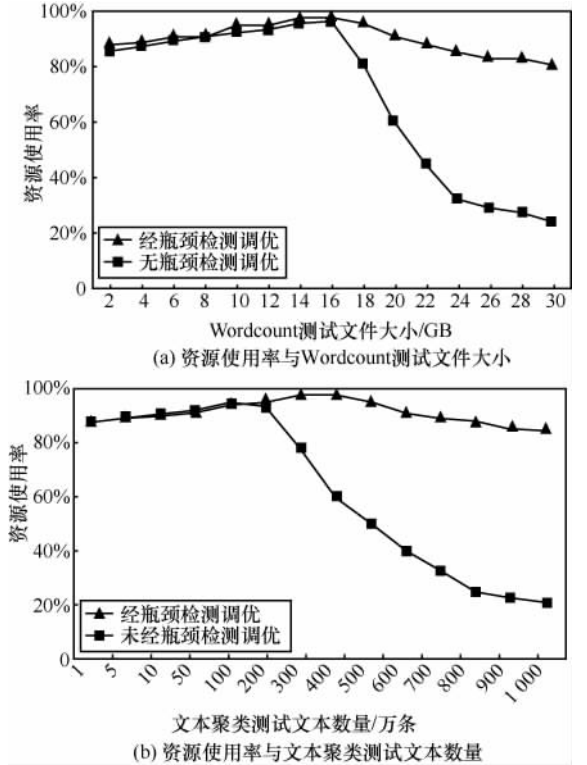


图 4 针对瓶颈检测出的资源进行调优前后对比

5 结束语

本文针对 Hadoop 分布式处理集群, 提出基于信息增益的瓶颈检测方法。该方法使用 RS 来评价集群性能。对集群作业 RS 和各资源使用的信息进行监控。然后计算各类资源增益, 确定系统的瓶颈资源。在确定集群瓶颈资源后, 便可针对该瓶颈资源进行相应调整措施以优化集群, 从而提高集群的性能。本文通过实验, 充分证明了其可以有效地检测 Hadoop 集群具体的瓶颈资源, 可以为集群的优化提供科学的指引和支持。

参考文献:

[1] 中国互联网络信息中心. 第 37 次中国互联网络发展状况统计报告 [EB/OL]. [2016-02-10]. http://www.cnnic.cn/hlwfzyj/hlwxzbh/hlwtjbg/201601/t20160122_53271.htm.
China Internet Network Information Center. The 37th China internet network development state statistic report [EB/OL]. [2016-02-10]. <http://www.cnnic.cn/hlwfzyj/hlwxzbh/hlwtjbg/201601/>



t20160122_53271.htm.

- [2] 张呈. Hadoop 集群海量小文件优化处理[D]. 武汉: 武汉理工大学, 2014.
ZHANG C. Mass small files to optimize processing under the Hadoop cluster [D]. Wuhan: Wuhan University of Technology, 2014.
- [3] 唐霞. Hadoop 调度器优化及其在舆情分析中的应用 [D]. 北京: 北京化工大学, 2015
TANG X. Hadoop scheduler optimization and its application in public opinion analysis[D]. Beijing: Beijing University of Chemical Industry, 2015.
- [4] 曾婉琳, 陈兴蜀, 罗永刚. Hadoop 节点资源参数优化策略[J]. 计算机工程, 2016(1): 1-6.
ZENG W L, CHEN X S, LUO Y G. Hadoop node resource parameter optimization strategy[J]. Computer Engineering, 2016 (1): 1-6.
- [5] 董新华, 李瑞轩, 周湾湾, 等. Hadoop 系统性能优化与功能增强综述[J]. 计算机研究与发展, 2013, 50(z2): 1-15
DONG X H, LI R X, ZHOU W W, et al. Hadoop system performance optimization and function enhanced review [J]. Journal of Computer Research and Development, 2013, 50 (z2): 1-15.
- [6] 李铎铭. 基于 MapReduce 性能优化的研究 [D]. 上海: 上海师范大学, 2015.
LI Y M. Based on graphs performance optimization research[D]. Shanghai: Shanghai Normal University, 2015.
- [7] DA V, PRADHAN P, DAN R. Provisioning servers in the application tier for E-commerce systems [J]. ACM Transactions on Internet Technology, 2007, 7(1): 57-66.
- [8] 朱显杰. 大规模复杂系统瓶颈检测和性能预测方法的研究[D]. 杭州: 浙江大学, 2010.
ZHU X J. Large-scale complicated system bottleneck detection and performance prediction method research [D]. Hangzhou: Zhejiang University, 2010.
- [9] GHEMAWAT S, GOBIOFF H, LEUNG S T. The Google file system[J]. ACM Sigops Operating Systems Review, 2003, 37(5): 29-43.
- [10] DEAN J, GHEMAWAT S. MapReduce: simplified data processing on large clusters [J]. Communications of the ACM, 2008, 51(1): 107-113.

[11] Apache Software Foundation. Eagle: secure Hadoop in real time [EB/OL]. [2016-02-11]. <http://eagle.apache.org/>.

[12] 王小巍, 蒋玉明. 决策树 ID3 算法的分析与改进 [J]. 计算机工程与设计, 2011, 32(9): 3069-3072.

WANG X W, JIANG Y M. Analysis and improvement of the decision tree ID3 algorithm[J]. Computer Engineering and Design, 2011, 32(9): 3069-3072.

[作者简介]



谭造乐(1990-), 男, 广东工业大学硕士生, 主要研究方向为社交网络数据挖掘、分布式架构。



郝志峰(1968-), 男, 广东工业大学教授、博士生导师, 主要从事机器学习、人工智能等研究工作。



蔡瑞初(1983-), 男, 广东工业大学教授、博士生导师, 主要从事数据挖掘、机器学习、信息检索等研究工作。

肖晓军(1970-), 男, 博士, 现就职于广州优亿信息科技有限公司, 具有多年电信行业管理工作经历, 主要研究方向为大数据、数据挖掘与电信行业应用等。

卢宇(1983-), 男, 广州优亿信息科技有限公司高级软件开发工程师, 主要从事大数据、机器学习与电信行业应用等相关研发工作。